TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

AHTI RUUSUVUORI

PROCESSING AND VISUALIZING WEBSITE BROWSING DATA FOR TARGETED CONTENT APPLICATIONS

Master of Science thesis

Examiner: Prof. Outi Sievi-Korte
Examiner and topic approved by the
Faculty Council of the Faculty of
Computing and Electrical Engineering
on 31st January 2018

ABSTRACT

AHTI RUUSUVUORI: Processing and visualizing website browsing data for targeted content applications
Tampere University of Technology
Master of Science thesis, 42 pages, 6 Appendix pages
October 2018
Master's Degree Programme in Information Technology
Major: Pervasive Systems
Examiner: Prof. Outi Sievi-Korte
Keywords: machine learning, data visualization, targeted content, website session data

Due to the rapidly increasing amounts of information on the Internet, providing users with relevant content becomes increasingly important. To achieve this, websites may employ recommender systems to offer targeted content to their users. In addition, due to the amount of information, recommender systems have begun adapting machine learning algorithms to more efficiently and accurately provide relevant content for the user.

Machine learning based approaches to recommender systems often involve "teaching" the system by feeding them pre-existing data about users and their preferences. This study aims to provide a machine learning based approach to a situation where a website has no specific data related to user preferences. Instead, the browsing patterns of previous users on a website are observed and analyzed to provide an estimation of the possible interests of new users.

The K-means clustering algorithm is used to evaluate the validity of clustering anonymous session data from a website visitor tracking system as a basis for identifying types of users. The clustered data is plotted into a scatterplot for analysis and used to examine the existence of clusters. In the case of distinct clustering, the clusters can be labeled as user groups, and further visitors can quickly be assigned to one of the groups to provide targeted content to.

K-means clustering is shown to perform suboptimally due to limitations in the algorithm's implementation as well as high amounts of intracluster noise in the source data. However, the data exhibits areas of density and sparsity and could potentially provide meaningful results with a different clustering algorithm.

TIIVISTELMÄ

AHTI RUUSUVUORI: Verkkosivun selailutietojen käsittely ja visualisointi kohdistetun sisällön ratkaisuja varten
Tampereen teknillinen yliopisto
Diplomityö, 42 sivua, 6 liitesivua
Lokakuu 2018
Tietotekniikan koulutusohjelma
Pääaine: Pervasive Systems
Tarkastajat: Prof. Outi Sievi-Korte
Avainsanat: koneoppiminen, datan visualisointi, kohdistettu sisältö, verkkosivun istuntodata

Johtuen nopeasti kasvavasta tiedon määrästä Internetissä, olennaisen sisällön tarjoaminen käyttäjille muuttuu yhä tärkeämmäksi. Tämän saavuttamiseksi verkkosivustot voivat käyttää sisällönsuositusjärjestelmiä tarjotakseen kohdistettua sisältöä käyttäjilleen. Lisäksi, johtuen tiedon määrästä, suositusjärjestelmät ovat alkaneet hyödyntää koneoppimisalgoritmeja tuottaakseen merkityksellistä sisältöä käyttäjälle tehokkaammin ja tarkemmin.

Koneoppimista hyödyntävät suositusjärjestelmt vaativat usein järjestelmän "opettamista", jolla käsitetään järjestelmälle jo olemassa olevaa käyttäjädatan syöttämistä. Tämä tutkimus pyrkii tarjoamaan koneoppimiseen pohjautuvan lähesty-mistavan tilanteeseen, jossa verkkosivustolla ei ole käyttäjistä tai käyttäjiltä kerättyä dataa. Tämän sijaan tutkimuksessa tarkastellaan ja analysoidaan edellisten käyttäjien selailutapoja verkkosivulla uusien käyttäjien kiinnostuksen kohteiden arvioimista varten.

Tässä tutkimuksessa käytetään K-means klusterointialgoritmia verkkosivun käyttäjäseurantarekisteristä saadun anonyymin istuntodatan klusteroitumisen tutkimiseen ja tarkastellaan sen pätevyyttä käyttäjäryhmien tunnistamiseen. Klusteroitu data piirretään pistekaavioksi analyysia ja klustereiden tunnistusta varten. Jos datasta voidaan tunnistaa selkeitä klustereita, ne voidaan luokitella käyttäjäryhmiksi, jonka jälkeen uudet käyttäjät voidaan nopeasti jaotella näin luotuihin ryhmiin ja heille voidaan tarjota ryhmälle kohdistettua sisältöä.

K-means klusterointialgoritmin osoitetaan suoriutuvan datan klusteroinnista huonosti johtuen algoritmin toteutuksen rajoitteista sekä datassa esiintyvästä korkeasta klustereiden välisestä kohinasta. Käytetyssä datassa esiintyy kuitenkin selkeitä tihentymiä sekä harventumia, mistä päätellen siitä olisi mahdollista johtaa merki-

tyksellisiä tuloksia toisenlaisilla algoritmeilla.

PREFACE

I would like to thank Jari Pohjosmäki from Abako Ltd. for the opportunity to help further the company product development with this study, as well as suggesting and aiding in shaping the idea behind it. I would also like to thank Petri Raivio and Marko Hautala from Abako Ltd. for providing valuable feedback during the study. Finally, I would like to thank Professor Outi Sievi-Korte from Tampere University of Technology for continuing to supervise and guide the work, even well beyond the original planned schedule.

Tampere, 22.10.2018

Ahti Ruusuvuori

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

API	Application Programming Interface
CDN	Content Delivery Network
CMS	Content Management System
DBSCAN	Density-Based Spatial Clustering for Applications with Noise
DOM	Document Object Model
GMM	Gaussian Mixture Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
JSON	JavaScript Object Notation
SSE	Sum of Squares Error
SVG	Scalable Vector Graphics
W3C	World Wide Web Consortium
WSS	Within-cluster Sum of Squares
XML	Extensible Markup Language

1. INTRODUCTION

The Internet continues to constantly grow as an increasing number of services and resources are integrated into an online environment. As the amount of content grows, the importance of user experience and ease of access becomes more significant. An essential part of a good online user experience is the availability and accessibility of desired information on websites. Due to this, content-heavy services such as Netflix [16] as well as social networking sites such as Facebook [36] have adapted recommender systems that offer users targeted content based on their preferences to avoid forcing the user to manually sift through irrelevant data.

According to Portugal et al. [32], recommender systems have recently begun to employ machine learning algorithms to aid in providing content the user is interested in. These algorithms often cross-reference certain attributes of data with user preferences or utilize content-based models that estimate the user's interests based on their activity. However, as explained by Sullivan [34], a fundamental part of machine learning techniques is training the algorithms with pre-existing data.

This study examines the possibility of using a data clustering algorithm to process and visualize a large number of previous visitors' session data that can then be analyzed to identify possible new groups of users. Due to the nature of the algorithm used, the source data must exhibit a clustering structure for the results to be meaningful. However, if the existence of definite clusters can be verified, it will prove clustering anonymous session data to be a viable way to identify new users on a website for targeted content applications.

1.1 Existing applications and study motivation

Since machine learning algorithms require statistical data to provide meaningful results, situations where a system has none can prove problematic. However, session data and website access logs are ubiquitous and present in most web servers [6]. These logs contain large amounts of information about users' actions on the website, such as link clicks and searches, and can be used to research browsing patterns.

One study, done by Poornalatha et al. [31], examines clustering of session data using a modified `K-means clustering` algorithm. This study explored the navigational patterns of users by clustering the accessed paths on a website. The results of the study could be used as a basis for recommender systems as well as a prediction system for websites that could reduce latency experienced by the site's user.

Another study done by Li et al. [24] also examines clustering of user's preferred browsing paths to improve upon existing website mining algorithm accuracy and efficiency. This study combines a modified clustering algorithm with an existing web mining algorithm and examines the size of the visited page, time spent on the page as well as the number of visits to the page as a representation of the page interest.

Despite the studies into mining and clustering user preferred website browsing, few studies examine clustering of session data together with website metadata as a basis for approximating new users' interests and offering targeted content. Looking into the possibility of utilizing an unsupervised machine learning technique and using ubiquitous session data could provide a means for quickly offering new users with relevant content despite the lack of statistics or user preferences.

Additionally, the implementation of the method is more simple than fully tailored recommender systems. While it does not offer the same accuracy as results based on user preferences, it could solve the issues caused by cold starts and offer a lightweight alternative to more complex solutions.

1.2 Research Methodology

This study is a case study that seeks to answer the question "Can cluster analysis be applied to analyzing session data as a basis for classifying users on a website?". Therefore, the study combines elements from machine learning, website personalization and metadata as well as data visualization.

According to Runeson et al. [33], case studies are an empirical research method that aim to study certain contemporary phenomena in their natural context. Case studies are a multidisciplinary method that can carry elements from both qualitative and quantitative research methodology. Thus, they can be applied to software engineering as well.

Runeson et al. continue to divide research methodologies into four types of purposes: exploratory, descriptive, explanatory and improving [33]. Exploratory purposes aim

to observe phenomena and generate new ideas or hypotheses based on what is observed, descriptive purposes aim to examine and describe phenomena, explanatory purposes seek to find an explanation to a problem and improving purposes seek to improve a phenomenon or its aspects. Based the original hypothesis and methods used, this study follows the exploratory purpose.

Finally, to maintain precision and credibility of empirical research, triangulation is needed. According to Runeson et al., there are four different types of triangulation to be considered: data, observer, methodological and theory triangulation [33]. These types refer to using more than one source of data, observer, method of research or theory in the study, respectively. Due to the nature of the study, the most relevant type is data triangulation. While the source data consists of the database of a single website, several different data sets are retrieved and studied.

The study has strong elements of quantitative research methods due to the ubiquity and statistical nature of the data used, but it is conducted as a case study with a single website in mind. However, due to the nature of the data and methods used, they can be applied to a wider range of website applications.

2. BASIC CONCEPTS

This study examines the relationship of various attributes in session data and website metadata to study their validity for identifying and categorizing website users. The parsing and processing of the data is achieved by utiziling a machine learning algorithm, and the results are visualized in a web environment for further analysis. Therefore, to understand the methods used in the study, it is necessary to examine the underlying concepts. This chapter explains the base concepts of website personalization and targeted content, session data, machine learning as well as vector graphics and presentation of data.

2.1 Website personalization and targeted content

A vast majority of content-rich websites, especially in e-commerce, incorporate a degree of personalization. According to Pappas et al., [28], website personalization can be viewed as the means of customizing and tailoring the site's content to better fit the needs and interests of its visitors.

As seen in a study by Flesca et al. [9], website personalization can have a noticeable effect on the website user experience. The study examined user preferences as well as page content as basis for tailoring the website navigation menu. The effects were studied on three modern websites, all of which showed improved user experience due to the personalization.

In addition to improved user experience, website personalization can be viewed from a business perspective. According to Jackson [20], a website that employs personalization is more likely to attract and prompt registration and service subscription from visitors. Due to this effect, businesses are seeking to maximize personalized customer interaction on their websites.

Implementation of personalization can be divided into three phases, starting from the most rudimentary customer recognition and adding layers of methods for more advanced personalization. The various stages of the implementation can be seen in table 2.1 [20].

Table 2.1 *Phases of personalization implementation [20]*

Phase I	**Phase II**	**Phase III**
Basic	Intermediate	Advanced
Customer recognition	Customer provided	Preference-based personalization
Mail or website addressed to the individual	User-provided information from surveys, registration forms, etc	Tracking user activity, comparing that activity with other users behaviour and predicting what the user would like to see next

Different companies may employ different phases of these implementations, depending on the nature of the services their website offers. The most relevant phase to this study is phase III, where the website collects and analyzes the users' activity and predicts their preferences based on the analysis. According to Jackson [20], the personalization process in this third phase can be divided into six steps or characteristics that define whether a website can be considered 'personalized':

1. identification,
2. data capture,
3. analysis and refinement,
4. match,
5. merge and delivery, and
6. optimization.

Identification stands for identifying the user and enables the following steps in the process. In the data capture step, all relevant data about the user is collected with various methods, such as tracking user activity on the site. This data is then analyzed and refined, which is often done in real-time.

After analysis, in the match step, the gathered data can be employed to craft user profiles. This can be done following various methods, such as *rules-based-matching* which crafts profiles based on user preferences and requested information, or *collaborative filtering*, which compares the profiles to various affinity groups constructed of profiles of other users. [20]

The user profiles hold information about the user's preferences, which is then used to cross-reference and deliver the user content relevant to their interests in the merge

and delivery step. The final optimization step includes repeating the previous steps as user interactions on the site increase. Repeating the steps allow for refining the created user profiles to improve their effectiveness.

Once these characteristics have been met, a site could be considered 'personalized'. As explained in section 1, some websites have begun utilizing recommender systems that automate this process. This study features the steps 2 and 3, or data capture and analysis and refinement respectively. The match step is replaced by visual analysis of the results of the clustering algorithm and identification of clusters.

The data used and the results produced in this study are anonymous, therefore the identification step is not relevant. Additionally, this study aims to produce results that could be employed by targeted content applications, therefore the merge and delivery as well as optimization steps are beyond its scope.

2.2 HTTP cookies and session data

Understanding website browsing data requires knowledge of what kind of data is being stored, how it is handled and what it is used for. In addition to user-specific and user-oriented data, websites and their pages contain metadata which describes various parameters and attributes of the page and its content.

Hypertext Transfer Protocol (HTTP) Cookies are a method for any website to store user-specific data and page related settings for future reference. They are stored onto a user's computer by the HTTP server and contain information about the user's actions on the website, allowing the server to maintain a stateful session with the user by reading and referencing the stored data.

The structure of a cookie comprises of six attributes:

- The name of the cookie
- The value of the cookie
- The expiration date
- The path the cookie is valid for
- The domain the cookie is valid for
- The secure attribute. This dictates the requirement for a secure SSL connection.

Cookies are used by most websites to store values relevant to the site, such as information about the user's browser or their session with the HTTP server. However, in some cases the website may store more personal information in cookies, if given by the user. The information is stored in the cookie-value parameter and can be dictated by the website.[2]

Cookies may be used for several different purposes. Due to the manner they are employed, cookies can be used by servers to recognize when a user returns to a web site or visits another website, for the purpose of tracking the user. Aside tracking cookies, they can also be used to temporarily store stateful session data between the user and the HTTP server. Session data is ubiquitous and often refers to anonymous information about the user's preferences, such as language. [2]

Session data may also contain information about the user agent, including but not limited to the name of the browser and type of device used to access the website as well as time of access and pages visited. As seen in the example extract of session data in appendix A, the stored data may also contain the Internet Protocol (IP) address as well as host name and domain name of the user.

The example of session data is extracted from the visitor tracking system of the website used in this study. While the data collected by the system is anonymous, it contains valuable information and metrics about the traffic on the site. As this type of data is readily accessible in many commercial websites [6], it can be used for analysis, which can help in maintaining and administering the systems, as well as, in some cases, provide insight for business related practises. This study utilizes the data for examining the interests of visitors on the website.

2.3 Website metadata

Associated meta-information of a website, contained in the `<meta>` tag of a Hypertext Markup Language (HTML) document, comprises of various information about the page and its contents. This information most commonly describes the content, its quality and attributes. It is not relevant to the user of the website, and is usually only read by the HTTP server.[3]

In addition to the HTML metadata, websites and search engines may employ proprietary means to attribute its content with specific keywords or tags for search purposes. This simplifies locating content relevant to a particular interest or subject. In addition, attributing pages with keywords will help users locate more content relevant to the subject on the website.

These keywords are utilized by both search engines such as `Google` [1] as well as website-specific, proprietary search functions, such as the search tool on `tampere.fi`, the website examined in the study. According to Lawrence [1], search engines operate by building databases and indexes of keywords attributed to various content, as well as the IP addresses of the websites the content is located on.

In the case of website-specific search tools, the search mechanism remains the same; content on the website is located by typing in keywords, which are then cross-referenced with those attributed to the content. In both cases, any content that matches the given keywords will be provided to the user as a search result.

For these search tools to provide reliable results, excluding contentless pages such as index pages and those dedicated to search results, all content on the website should be attributed with keywords. In the case of this study, these keywords are cross-referenced with traffic on the website to explore correlation.

2.4 Machine learning and cluster analysis

As described by de Mello et al. [8], machine learning is a field of computer science that studies methods to enable computers and software to "learn" to perform tasks without being explicitly programmed. It is a subset of the study of artificial intelligence and is based on statistical learning theory. Machine learning algorithms utilize pre-existing data to progressively improve upon the results of a given task, allowing for increasingly accurate results.

According to Sullivan [34], there are several different techniques of machine learning for various purposes, such as supervised learning, unsupervised learning, reinforcement learning and neural networks. Consequently, there are numerous practical applications for machine learning in other fields, including medical research [15], agriculture [19], stock markets [5] and weather prediction [21].

Of the aforementioned concepts in machine learning, the two most prominent classifications are supervised and unsupervised learning. Supervised learning methods assume that some degree of the desired results is already known beforehand and can be used to "train" the system, whereas in the case of unsupervised learning, the methods allow for producing meaningful results without outside influence. [34]

The most relevant method of machine learning for this study is cluster analysis. According to Sullivan, cluster analysis is an unsupervised learning technique, which employs various algorithms that partition multivariate data into distinct clusters

without pre-existing specifications or data classification. Cluster analysis can be used to discover patterns in the data to further identify instances and occurrences of individual events or attributes. For example, visualizing the relationships between the different species of the flower *Iris* and the effects of clustering them can be seen in figure 2.1.

Figure 2.1 *Effects of clustering*

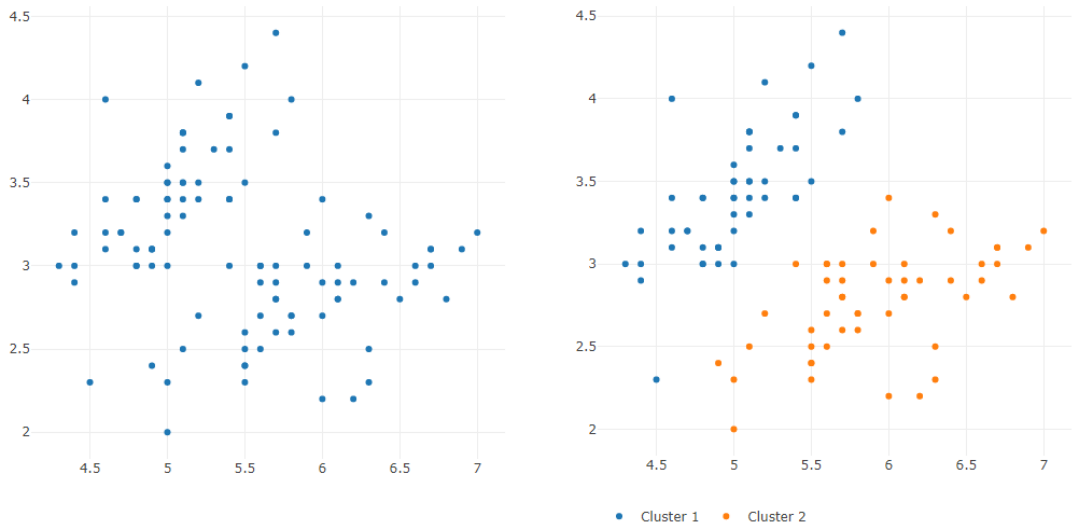


Figure 2.1 demonstrates how certain types of attributes may correlate with each other, forming distinct clusters in the data. With the help of clustering algorithms, such as the `K-means clustering` algorithm in this case, the clusters may be refined and verified, providing means of identifying and classifying previously unknown instances in the data.

Similar to machine learning as a whole, cluster analysis has several different applications in various fields. According to Hrdle et al. [18], the potential for classification of individuals based on attributes shared with the general population is useful in multiple fields such as natural sciences, medical sciences, economics and marketing.

Furthermore, cluster analysis in the field of computer science can be adapted to a wide range of applications, such as data mining [41][24] and network analysis [43]. Additionally, it can be utilized as a means of producing statistical data for content recommender systems.

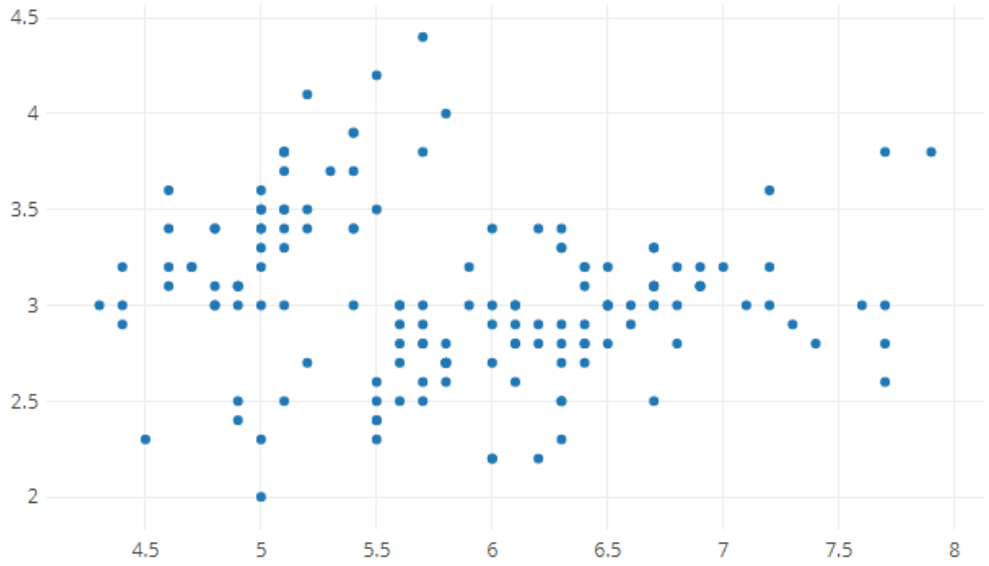
2.5 Vector graphics and presentation of data

Visualizing data is a fundamental part of data analysis. As described by Lea et al., ”data visualization is designed to decode and present complex data in a pictorial or graphical format enabling the decision-maker to clearly grasp difficult and esoteric concepts/ideas” [23, p. 923]. In other words, presenting results of a study or report in a clear and cohesive manner allows for better readability and reference, making it easier for a human to distinguish the most relevant information quickly.

Depending on the data and the nature of its context, it can be displayed in a number of different formats ranging from traditional graphs, plots and diagrams to flowcharts and density maps. In machine learning and especially cluster analysis, due to the multivariate nature of the data, the most suitable format for presentation is scatterplots.

According to Dinov [7], scatterplots are suitable for presenting bivariate relationships between data. An example of a bivariate scatterplot can be found in figure 2.2. The source data is constructed from the first two dimensions of a multivariate data sample of the attributes of the Iris flower [10].

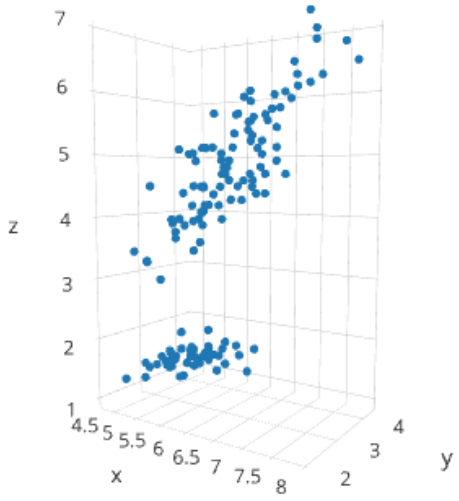
Figure 2.2 *Example of a bivariate scatterplot*



In these cases, the plot consists of two axes, each representing a variable, and the

data in the set is visualized as separate data points on the plot. This can, however, also be expanded into the third dimension by adding a third axis for a third variable, resulting in a three dimensional scatterplot, as seen in figure 2.3. The source data is constructed from the first 3 dimensions of the Iris data set [10].

Figure 2.3 *Example of a multivariate scatterplot*



Additionally, there are instances where the data may require more than three dimensions, such as the complete Iris data set. There are methods such as subspace and projection clustering [25] to present this type of data in a more readable format, however this study will concentrate on bivariate data sets and their representations.

When rendering a plot or diagram in a web environment, vector based graphics are often chosen instead of raster graphics to easier produce structured images. As opposed to raster graphics and bitmaps, vector graphics consist of geometric primitives that define lines, curves and other basic figures and therefore are not resolution-dependent. [27]

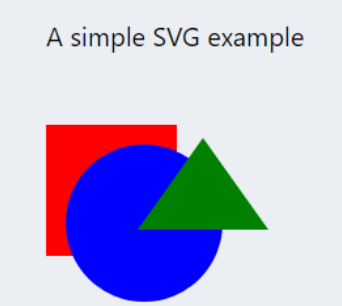
Furthermore, in a web environment, vector graphics can be implemented with the Scalable Vector Graphics (SVG) standard, a web graphics standard developed by the World Wide Web consortium (W3). It is royalty-free and vendor neutral, and therefore widely applied in the web environment. As described by Neumann, the standard is based on the Extensible Markup Language (XML) and utilizes a combination of vector graphics, raster graphics, scripting and animation. [27]

Due to these features, graphics rendered with the standard are freely scalable while retaining render quality. Additionally, as the standard is based on XML, it "integrates well with modern development tools and common workflows" [27, p 1831] as well as many web technologies, Neumann explains. In this study, due to its versatility and good integration, SVG is utilized for the presentation of data.

The SVG standard follows a tree-like document structure that describes its components with the XML based syntax. This means that the elements in a SVG document are declared with XML tags, e.g. the `<rect>` tag would declare a rectangle shape.

These tags can be given different attributes to manipulate its appearance or behaviour. Furthermore, due to its tree-like structure, elements can be nested inside each other, allowing for rendering of more complex shapes and patterns. An example of a simple SVG document embedded into a web page can be found in figure 2.4.

Figure 2.4 *A simple SVG document [27]*




```
<div class="svg" id="svg">
    <?xml version="1.0" encoding="UTF-8" standalone="no"?>
    <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
    "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
    <svg xmlns="http://www.w3.org/2000/svg"
    xmlns:xlink="http://www.w3.org/1999/xlink" id="flag-icon-css-cn"
    width="640" height="480">
        <text x="50" y="40" font-size="20">A simple SVG example</text>
        <rect x="50" y="100" width="100" height="100" fill="red"/>
        <circle cx="125" cy="175" r="60" fill="blue"/>
        <polygon points="120,180 170,110 220,180" fill="green"/>
    </svg>
</div>
```

One of the applications of SVG in the web environment is the Data-Driven Documents (`D3`) JavaScript library. The `D3` library offers means for manipulating HTML documents using web standards such as SVG [4]. It is capable of rendering SVG documents into the World Wide Web Consortium (W3C) HTML Document Object Model (DOM) while seeking to simplify the use of the DOM Application Programming Interface (API).

Other, similar JavaScript libraries have been developed that utilize the `D3.js` library. These libraries are usually built to refine the available functions of their parent library towards a more specific purpose. The library used in this study, `Plotly.js`, is an example of this kind of library.

3. TOOLS AND METHODS

3.1 Website visitor tracking systems and Apache Solr

Websites may utilize customized and proprietary systems for collecting data of site users and visitors. The contents and nature of the collected data varies depending on the website; According to Colombo [6], online retailers and other e-businesses are able to record a variety of data ranging from various actions and transactions to visiting patterns on the site. He continues to explain how the website servers are able to log every view, click and action taken on any page. However, this type of log data is large in quantity and may be difficult to extract meaningful information from.

In this study, the data used is extracted from a visitor tracking system on a large website with millions of yearly visits. The system is built on the Apache Solr open source search and indexing platform, which also allows for indexing website metrics and session data for future reference. The Solr platform can be configured to contain a number of index *cores*. According to the Apache Solr website, these cores refer to a single index instance with an associated transaction log and configuration files [12]. In other words, a number of different cores can be configured to index specific content or types of data.

In this case, one of the cores on the website's indexing system has been configured to collect and index data related to visitors' sessions based on their session cookies. Indexed data entries are stored in JavaScript Object Notation (JSON) objects, and consists of key-value pairs of various metrics such as information about the visitor's user agent and related website metadata. A more detailed example of the data can be found in appendix A. The example has been extracted from the test environment of the `tampere.fi` website used in this study.

3.2 The K-means clustering algorithm and Apache Ignite

According to Wu [42], there are five types of clustering algorithms: prototype-based, density-based, graph-based and hybrid algorithms as well as algorithm-independent

methods. `K-means clustering` is a prototype-based algorithm, where the prototype of a cluster is the *cluster centroid*. Centroids are an arbitrary point in a data set that denotes the center point of a distinct clustered structure within the data set. They also define the size and boundaries of a cluster based on the distance from any given data point to the nearest cluster centroid.

The `K-means clustering` algorithm provides a lightweight and robust solution for cluster analysis. However, there are cases where the algorithm may perform poorly, the most relevant of which is the lack of a clustering structure in the source data. As explained by Wu [42], a data set with non-globular clusters will cause unreliable results, therefore the validity of the clusters and their centroids produced by the algorithm will be evaluated visually.

The process of the algorithm follows a simple, iterative pattern. First, an initial value of K is chosen for the desired number of clusters. Then, every data entry in the set is assigned to a cluster based on its current nearest centroid. Once the whole data set has been iterated, the centroid locations are adjusted based on the mean distance of all the members of their corresponding clusters. After this, the data entries are re-assigned to the clusters, and the process repeats until data entries no longer change clusters or the value of new information is marginal. A pseudo code representation of `K-means` can be found in algorithm 1, as described by X. Jin et al. in their study [22].

Algorithm 1 K-means clustering [22]

Require: K, number of clusters; D, a data set of N points
Ensure: A set of K clusters

```
1: Initialization
2: repeat
3:     for each point p in D do
4:         find the nearest center and assign p to the corresponding cluster
5:     end for
6:     update clusters by calculating new centers using mean of the members
7: until stop-iteration criteria satisfied
8: return clustering result
```

An example implementation of the algorithm used for this study can be found in the Apache Ignite platform's collection of source codes for various platform specific applications. The algorithm is implemented in Java and utilizes the Apache Ignite platform and its distributed processing capabilities.

According to the product website, Apache Ignite is an open source platform for dis-

tributed workloads, databases, caching and processing that also employs workload streaming [11]. The algorithm is implemented in a manner that employs the platform's computing grid feature by launching a number of *Ignite nodes* upon execution that divide the workload amongst the available nodes. In addition, the algorithm was modified to accept the initial value of K as a parameter and calculate the sum of squares error (SSE), as well as read and write its data into external files.

Apache Ignite was chosen for this study due its distributed computing capabilities as well as offering an implementation of the algorithm that allows distributed computing. The data sets examined in the study are small enough to be calculated on a single node in feasible time, however, the performance of classic `K-means` and other clustering algorithms will decrease with large-scale data [42]. While distributing the workload among multiple computing nodes does not solve the performance issues of the algorithm, it would solve the limitations in processing power to run it.

3.3 JavaScript graphing libraries and plotly.js

As explained in section 2.5, in a web environment, graphing may be done with the help of JavaScript libraries specialized in graphing and data visualization. These libraries offer a selection of functions and methods for visualizing data quickly and efficiently. The resulting graphs can then be used to examine if the clustering of anonymous session data can be used to analyze and identify specific user groups for targeted content applications.

These libraries are often built in a manner that allows for visualizing the desired data with a simple declarative syntax [29][37]. In the case of `plotly.js`, graphs are generated based on a simple JSON object schema, as seen in program 3.1. The graph is rendered onto a HTML document via a function call in JavaScript that takes the JSON objects as parameters.

Program 3.1 *Plotly chart data with scatter plot [29]*

```
data = [
    {
        x: [1, 2, 3],
        y: [3, 1, 6],
        type: 'scatter',
        name: 'Name of trace',
               mode: 'markers',
        marker: {
            size: 3,
            color: 'rgb(16, 32, 77)'
        }
    }
];
```

As seen in the example, the data points are presented in separate arrays of integers, one for every axis. Additional attributes may be given to dictate the type and name of the trace, as well as the type and appearance of the data values.

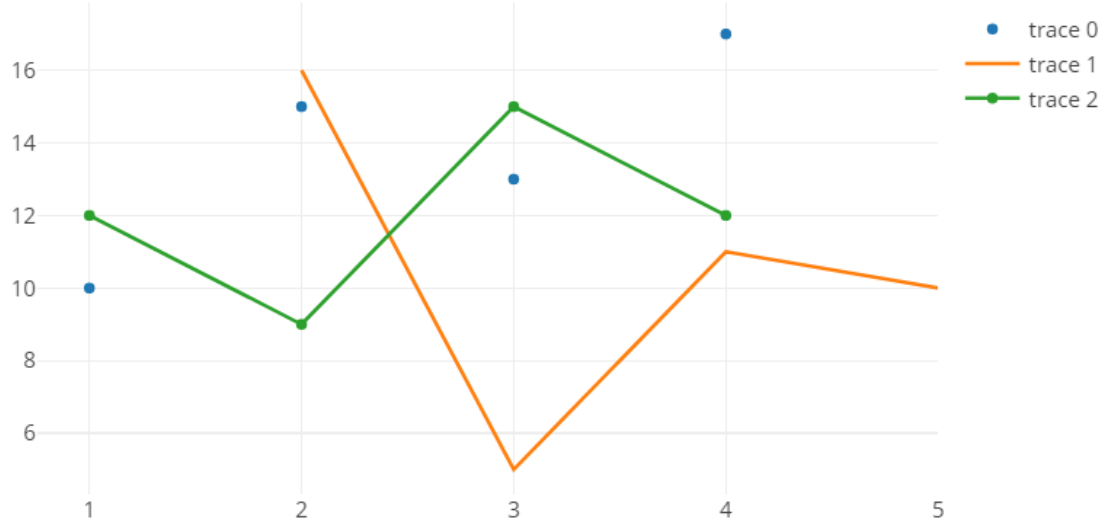
Another attribute the library accepts as a parameter is the chart *layout*. Following a similar JSON schema, it contains attributes for formatting the overlaying chart. It enables for example captioning the chart and its axes, dictating the chart dimensions and layout as well as assigning legends, labels and annotations. An example of the *layout* attribute can be found in program 3.2.

Program 3.2 *Plotly chart layout [29]*

```
layout = {
    title: 'simple example',
    xaxis: {
        title: 'time'
    },
    showlegend: true,
        legend: {
               orientation: "h"
        },
}
```

The `plotly.js` library allows for multiple traces of data to be rendered onto a single graph, each of which can have differing data as well as data type. Assigning multiple data sets into different traces makes it easier to distinguish relevant information from the same graph. For example, plotting three simple traces with the types "marker", "lines" and "lines+markers" would result in a graph that features all three layered atop each other, as seen in figure 3.1 [29].

Figure 3.1 *Sample scatter plot with plotly.js [29]*



Depending on the library, it can be utilized in a project or research by simply including the library in the source files, from where its functionalities can be invoked. In the case of `plotly.js`, the library can be employed either by including it as an external script via a Content Delivery Network (CDN) or downloading it from the GitHub repository [30]. In this study, the library was downloaded from the GitHub repository and included as an external script file.

4. IMPLEMENTATION

This chapter explains the various methods taken to produce the results. It describes the collection of the source data with a clientside script implemented in JavaScript, which will extract the relevant key-value pairs from the visitor tracking system and save the values in a two dimensional array. After this, the script transforms the values into integers, producing an data structure suitable for the clustering algorithm.

The `K-means clustering` algorithm is used to cluster the processed data and calculate the cluster centroids. After the data has been clustered, it is rendered onto a scatterplot with `plotly.js`, an open source JavaScript library for vector graphics, based on `D3.js`[30].

Finally, the rendered plot will be used to verify the existence of clusters in the data set. Any clustering present in the resulting data set can be identified as types of users on the website, and the existence or absence of these clusters will validate whether this method is suitable for processing data for targeted content applications.

4.1 Collecting and parsing the data

As explained in section 3.1, the website search and indexing platform contains an index core for indexing session data, stored in key-value pairs in JSON format. A truncated example query result can be seen in program 4.1. For the purpose of this study, there are three key-value pairs that hold the most significance: the path of the visited page, time of access and the device used.

Program 4.1 *Sample query result from the* ***tampere.fi*** *website session index*

```
{
        [...]
        "HitDate":"2016-04-11 15:50:25",
        "HitTime":1460379025710,
        "UA.OS.Name":"Windows",
        "UA.Browser.Name":"IE",
        "UserAgent":"Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1;
            ↪ Trident/6.0) Image size by Siteimprove.com",
        [...]
        "STATO_FULLPATH":"/channels/public/teksti/tampere/fi/index/
            ↪ tampereenkaupunki/ajankohtaista/tapahtumat/PYWirzN5h",
        [...]
        "UA.Device.Type":""
}
```

The website uses keywords to attribute certain content on its pages, which act as both a description of the content as well as a pointer for what the user visiting might be interested in. Every page of the site can be assigned keywords, however in this case, the keywords exist in a separate Solr index core. This second core contains the index for the content of the website, including the keywords for every relative path of the website. An example query result from the content index core can be seen in program 4.2. The result has been similarly truncated to the fields most relevant to the example.

Program 4.2 *Sample query result from the* ***tampere.fi*** *website content index*

```
{
        "StatoType":"tampere.tapahtuma",
        [...]
        "asiasanat":"musiikki,festivaalit,rock,populaarimusiikki,hip hop",
        [...]
        "PublicURL2":"https://www.tampere.fi/teksti/tampereen-kaupunki/
            ↪ ajankohtaista/tapahtumat/PYWirzN5h.html.stx",
        [...]
        "NavigationTitle":"Lost In Music",
        [...]
        "STATO_FULLPATH":"/channels/public/teksti/tampere/fi/index/
            ↪ tampereenkaupunki/ajankohtaista/tapahtumat/PYWirzN5h"
}
```

The relative path of the visited page, found with the key *STATO_FULLPATH*, is present in both index cores, and can be used to cross-reference the session data with relevant keywords, found with the key *asiasanat* in the content index core.

Therefore, in addition to retrieving the website sessions from the session index core, a query to the content index core is made for every result returned this way. The path in the query results from the session index core is used as the filter for the secondary query. The results for this query contain the keywords for the wanted path, which are then collected for further processing.

The time of access is relevant in deducing whether the time of day has effect on the accessed content. Finally, the information about the device used to access the page can give us information about how and where users access the website, and whether it has impact on what content they are looking to access.

The `K-means clustering` algorithm used to process the data requires a two dimensional array of integers as input, therefore it must be preprocessed into the correct format. JavaScript is used to pick the relevant fields from the data and translate the values into integers as necessary. The script will also create suitable data structures of the translated values; the resulting structure will be a matrix of integers suitable which is written into a file, which is then read in the clustering algorithm. A sample of the data structure can be found in program 4.3.

Program 4.3 *Input data for K-means clustering*

```
data = {
    {0, 0.0, 0.0, ... 0.0},
    {0, 0.0, 0.0, ... 0.0},
    {0, 0.0, 0.0, ... 0.0},
    ...
}
```

Each separate data entry will be represented as a single dimensional array of unsigned or floating point integers, which can be translated back into their original values for easier reference after clustering and visualization. The values are the time of access and accessed keyword, respectively.

The type of device used to access the site is considered as well, however, in this case, the session data extracted from the visitor tracking system only contains three possible values for device types: stationary, mobile and tablet. Therefore, instead of introducing a third dimension to the data set, it is divided into three different sets for easier visualization and examination.

Due to the implementation of the clustering algorithm, the first value of the arrays is reserved for an estimation of the cluster the entry belongs to, however in this case, it is arbitrary and can be assigned the value `0`. Finally, the arrays of entries are

collected into another array, transforming it into a two dimensional data structure suitable for the clustering algorithm. A more accurate description of the process of the script can be found in algorithm 2.

Algorithm 2 Conversion of data

Require: S, a source array of N entries
Ensure: D, a data set of N points

1: Initialization
2: **for** each point p in S **do**
3: push value *0* to array *arr*
4: find index of keyword, push to array *arr*
5: get time of day in seconds from key "HitTime", push to array *arr*
6: push *arr* to new array D
7: **end for**
8: **return** data set D

The time of access can be deduced from several different key-value pairs. In this case, the key *HitTime* is chosen, due to its values being in the *Unix epoch* format. The *epoch* is a format that displays the number of seconds that have passed since the first of January, 1970 [39], and contains the required information in integer format for constructing an exact date and time of day.

While the epoch is an integer, its values are relatively high. The values can be passed onto the clustering algorithm as they are, however for the `K-means clustering` algorithm to produce valid results, the coordinates given must be within the same scope. Therefore, the epoch values are first translated into JavaScript Date objects, from which the respective values for hours, minutes and seconds are extracted.

These values are converted into seconds to translate them into suitable format for clustering, however the resulting values are in the order of tens of thousands. Therefore, due to the amount of keywords being in the order of hundreds, the time of day is divided by one hundred (100) to produce the final value for the array. This way the relevant information remains and coordinates are normalized. The implementation in JavaScript can be seen in program 4.4.

Program 4.4 *Parsing time of access*

```
var data = JSON.parse(sourceData);

for (let i = 0; i < data.length; i++) {
        let HitTime = parseInt(data[i].HitTime);
        let HitTimeDate = new Date(HitTime);

        let HitHours = HitTimeDate.getHours();
        let HitMinutes = HitTimeDate.getMinutes();
        let HitSeconds = HitTimeDate.getSeconds();

        let xAxis = (HitHours * 60 * 60 + HitMinutes * 60 + HitSeconds)/100;
        [...]
}
```

The resulting values can be turned back into time of day for analysis by reversing the algorithm. After multiplying the value by one hundred (100), it is stored in a variable, which can then be used to calculate the minutes and hours. An example of this process can be seen in program 4.5.

Program 4.5 *Parsing time of access*

```
let secondsTotal = xAxis*100;
let seconds = secondsTotal\%60;
let minutesTotal = Math.floor(secondsTotal/60);
let minutes = minutesTotal\%60;
let hours = Math.floor(minutes/60);
let timeOfDay = hours+":"+minutes+":"+seconds;
```

First, to transform seconds into minutes and hours, the seconds are divided by sixty (60) to transform them into minutes. The remainder of the division will be the value for seconds in the time of day, which is stored into the *seconds* variable with the modulo (%) operator. The same procedure is done for the total amount of minutes to find the total amount of hours respectively. From the resulting variables, the time of day can be constructed by concatenating each value with a colon separator.

In the case of keywords, the number of different keywords per path is unknown at the time of parsing the data. Therefore, all the possible keywords in the source data are collected into an array. The resulting array and its indexes can be referenced when translating the keywords into integers and vice versa, allowing for humans to better distinguish the end result in the analysis phase. A sample implementation of constructing the array can be seen in program 4.6.

Program 4.6 *Parsing keywords*

```
for (let i = 0; i < data.length; i++) {
    [...]
    var keywordArray = [];

    // Keywords of single entries
    let Keywords = data[i].Keywords.split(",");

    // Create map for keywords
    for (let k = 0; k < Keywords.length; k++) {
        if (Keywords[k] != '' && jQuery.inArray(Keywords[k], keywordArray) ===
            ↪ -1) {
            keywordArray.push(Keywords[k]);
        }
    }
    [...]
}
```

The keywords retrieved from the Solr queries are contained in a similar key-value pair, where the value lists each relevant keyword in a string, separated by a comma. The list is iterated and each unique instance of a keyword is pushed into the keyword array, resulting in a data structure that contains all possible keywords. These keywords can be retrieved with and referred to by their index, allowing for quick translation between their readable format and respective integer values.

Once the array of keywords has been created, the separate entries are collected into a two dimensional array, which is then written into a text file to be read in the clustering algorithm. JavaScript supports creation of files in the client with the JavaScript `Blob` object, which is used to handle data that is not necessarily native JavaScript format [26].

After creating the `Blob` object, the browser can be made to prompt the user to download the object as a file. However, a more simple way of achieving the same result, as used in this study, is to display the data on an otherwise blank webpage and save its contents manually from the browser. This method also ensures native line breaks are maintained.

The implementation of the clustering algorithm expects the data to be in a specific format within the file. The format follows a simple matrix structure, where the values of a single entry are separated by a space, and different entries are separated by a line break. A truncated example of the formatting can be seen in program 4.7.

Program 4.7 *Clustering source data format*

```
0 216,11 0
0 146,22 0
0 265,76 0
0 146,68 0
0 97,08 0
...
0 340,81 767
```

As explained in section 4.1, the first integer of each entry is arbitrary, and therefore has been assigned the value 0. The following floating point integer is the time of day in seconds, which has been normalized by dividing it by one hundred (100). The last integer is the index of the keyword. Once the file has been created, the data is ready to be clustered.

4.2 Clustering the data

The `K-means clustering` algorithm used is implemented in Java. An example implementation of the algorithm from the Apache Ignite platform's source code library was used and modified to better fit the purposes of this study. The full, modified source code can be found in appendix B.

The source data for the clustering is read from an external file, the creation of which was explained in chapter 4.1. A two dimensional array is constructed from the source data while maintaining its structure. The first variable in each entry is the time of day and the second is the index of the related keyword. After the data is parsed and a suitable data structure is formed, the data is read into the cache memory, from where it is retrieved for the clustering for optimal performance. An algorithm for parsing the data can be found in program 4.8.

Program 4.8 *Preparing data for clustering*

```
double[][] data = new double[0][0];
Pattern spacePattern = Pattern.compile(" ");
try {
    data = Files.lines(Paths.get("/* Path to file */"))
        .map(item -> spacePattern.splitAsStream(item).mapToDouble(Double::
            ↪ parseDouble).toArray())
        .toArray(double[][]::new);
}
    catch(IOException e) {
    e.printStackTrace();
}
```

In addition to the source data, the `K-means clustering` algorithm requires an initial estimation of clusters present in the data to produce reliable results. To estimate the number of clusters centroids in the data, the `elbow method` is used. This method involves running the algorithm with a range of values of K, starting from `K=1` and incrementing by 1, until such a value is found that intracluster variation is sufficiently small and incrementing further would not add more valuable information.

As explained by Syakur et al. [35], the `elbow method` consists of plotting the within-cluster sum of squares (WSS), also known as sum of squares error as a function of the number of clusters. From the resulting graph, a number of clusters is chosen where the amount of margin in intracluster variance drops, forming the "elbow".

In the case of clustering, as shown by Hansen et al. [17], SSE is calculated by adding together the squared distances from the cluster centroids to its respective data points:

$$SSE(C_j) = \sum_{i=1}^{n} (x_i - \overline{x}_j)^2 \tag{4.1}$$

where C_j is the current cluster, x_i is the distance of data point i and $\overline{x}_j$ is the mean of all distances within the cluster.

The example algorithm provided by the Apache Ignite platform does not provide an implementation for calculating SSE, therefore the algorithm must be modified so that SSE may be found. First, to form a data structure to hold the clusters and their distances, a two dimensional list is created and populated with empty lists of doubles for every value of K. Additionally, an *EuclideanDistance* object for calculating the distances within the cluster is declared, as described in program 4.9.

Program 4.9 *Map of cluster centroids and the distances to its datapoints*

```
List<List<Double>> clusterMap = new ArrayList<List<Double>>();
for (int i = 0; i < K_value; i++) {
    clusterMap.add(new ArrayList<Double>());
}

// Initialize distance variable for calculating SSE
EuclideanDistance distance = new EuclideanDistance();
```

Next, the data set is iterated as per normal process of the algorithm. During each iteration, a prediction that assigns data points into clusters is made. Additionally, the distance between the cluster centroid and the current data point is calculated and added into the cluster map created earlier. Finally, the iteration saves the data points and the indexes of their respective cluster centroids into ArrayLists to be later written into an output file, as seen in program 4.10.

Program 4.10 *Clustering and distance calculation*

```
for (Cache.Entry<Integer, double[]> observation : observations) {
    double[] val = observation.getValue();
    double[] inputs = Arrays.copyOfRange(val, 1, val.length);

    // Assigning data points to clusters
    double prediction = mdl.apply(new DenseLocalOnHeapVector(inputs));

    // Calculate distance between datapoint and its assigned cluster centroid
    int clusterIndex = new Double(prediction).intValue();
    double ret = distance.compute(mdl.centers()[clusterIndex],inputs);

    // Assign distance to the clustermap
    clusterMap.get(clusterIndex).add(new Double(ret));

    // Output arraylists
    xAxis.get(clusterIndex).add(new Double(inputs[0]));
    yAxis.get(clusterIndex).add(new Double(inputs[1]));
}
```

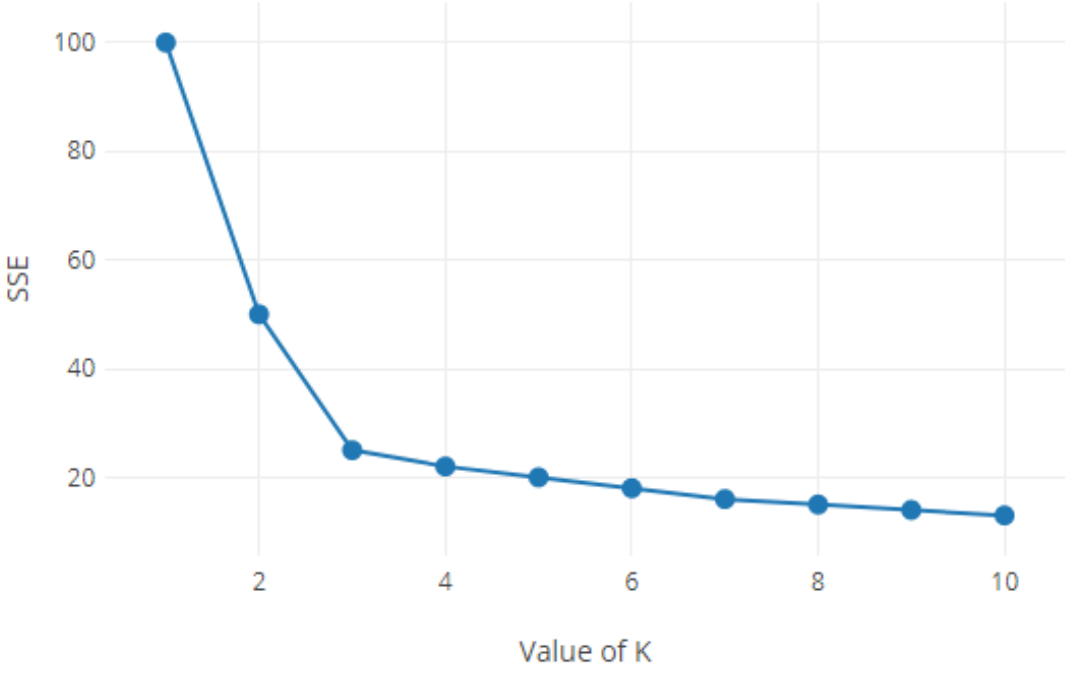
Once the necessary data structures and objects have been declared and populated with the suitable data, SSE can be calculated. The algorithm, as seen in program 4.11, iterates the *clusterMap* data structure and calculates the average of all distances within each cluster. The sum of squares is then calculated by iterating each list of distances of each cluster and comparing them to their respective average, before squaring the results and adding them together.

Program 4.11 *Calculating SSE*

```
double sum = 0;
double average;
double[] sse = new double[K_value];

for (int i = 0; i < clusterMap.size(); i++) {
    sum = 0;
    sse[i] = 0;

    for (double d : clusterMap.get(i)) sum += d;
    average = sum / clusterMap.get(i).size();

    for (int j = 0; j < clusterMap.get(i).size(); j++) {
        sse[i] += Math.pow(clusterMap.get(i).get(j) - average, 2);
    }
    System.out.println("Cluster "+i+" SSE: "+sse[i]);
}
```

The resulting values are the within-cluster sums of squares, which then need to be added together to form the total sum of squares. The final result is then plotted against the value of K, forming a graph suitable for examination, an example of which can be seen in figure 4.1.

Figure 4.1 *Example of SSE*



In an ideal situation, a clear ”elbow” can be seen in the graph. In such a case,

the elbow can be found at a point at which the margin in intracluster variance is reasonably low and exhibits little change as the value of K is increased. Therefore, in this example, the optimal value for K and the number of clusters to examine would be three (3).

However, there are situations where the `elbow method` cannot provide unambiguous results. If the clustered data set does not have a clustering structure, the drawn graph will resemble a curve rather than exhibit any clear angle [35]. In this case, a number of clusters is chosen such that the margin will be reasonably low.

After SSE has been calculated and the data clustered, the results are written into an external output file. For this purpose, two *ArrayLists* have been created, which were populated with the clustering results in program 4.10. The declaration and initialization of the ArrayLists is shown in program 4.12.

Program 4.12 *Declaring ArrayLists for results*

```
// Arraylists for output
List<List<Double>> xAxis = new ArrayList<List<Double>>();
for (int i = 0; i < K_value; i++) {
    xAxis.add(new ArrayList<Double>());
}
List<List<Double>> yAxis = new ArrayList<List<Double>>();
for (int i = 0; i < K_value; i++) {
    yAxis.add(new ArrayList<Double>());
}
```

Due to the requirements of the visualization component of this study, the clustered results are split into two different ArrayLists, one for each axis. To create an output file with the results, the contents of the ArrayLists are iterated and written into a new file using the Java *FileWriter* class, as seen in program 4.13.

Program 4.13 *Writing the output file*

```
try (FileWriter xAxisFile = new FileWriter("/* Path to file *//xAxis.js")) {
    for (int i = 0; i < xAxis.size(); i++) {
        xAxisFile.write("var xAxis"+i+" = "+xAxis.get(i)+";\n");
    }
} catch(IOException e) {
    e.printStackTrace();
}
try (FileWriter yAxisFile = new FileWriter("/* Path to file *//yAxis.js")) {
    for (int i = 0; i < yAxis.size(); i++) {
        yAxisFile.write("var yAxis"+i+" = "+yAxis.get(i)+";\n");
    }
} catch(IOException e) {
    e.printStackTrace();
}
```

The resulting files will be JavaScript files which contain arrays of clustered coordinates. The coordinates are divided into different arrays based on the number of clusters and assigned into JavaScript variables, which can then be included and read directly from the files into the visualization component. Dividing the coordinates into cluster-specific variables allows for coloring and manipulating them as separate traces during visualization.

4.3 Creating a visualization tool with plotly.js

The graphs used to visualize the clustered data are rendered onto an HTML document with the help of the `plotly.js` library, allowing them to be viewed in a standard web browser. At its core, the HTML document requires a HTML DOM object for every separate graph to be rendered. In this case, a `<div>` element with an id attribute is sufficient.

An added client-side script selects the element with the DOM object's *querySelector()* function, for which the `plotly.js` library's *plot()* function is then called. The *plot()* function acts as an API between the library's functionalities and the rest of HTML document.

As explained in section **??**, the *plot()* function, that renders the data onto the HTML document, expects its parameters to be arrays of JSON objects as well as follow a specific format. Due to this, a trace object template was created for every separate cluster, as seen in program 4.14.

Program 4.14 *Trace of a single cluster*

```
trace_n = {
        x: xAxis_n,
        y: yAxis_n,
        name: 'Cluster n',
        mode: 'markers',
        type: 'scatter',
        marker: {
                size: 3
        }
}
```

The arrays for the x and y coordinates are read from variables, which are included in the external files provided by the clustering algorithm. Dividing the different clusters into separate traces for the plotting allows for assigning different colors to each cluster, resulting in a graph where the clusters can be easily distinguished by the human eye.

Once the traces and a basic graph layout have been created, they are passed to the *plot()* function as parameters, along with the target DOM object. An example of the function call can be found below in program 4.15.

Program 4.15 *Plotly chart rendering*

```
this.querySelectorAll('.scatterPlot').forEach(function (elem) {
        var id = elem.id;
        Plotly.plot(id, data, layout);
});
```

In this example, all suitable DOM objects are selected from the document and the *plot()* function is called for all of them. This method allows for multiple instances of the graph to be rendered, although for the purpose of this study, only one is needed. The `plotly.js` library will read the parameters given to it and render a graph onto the specified DOM object.

4.4 Identifying new user groups

The resulting graph on the HTML document has additional features for further manipulating and analyzing the plotted data. These features include toggling the visibility of the data traces as well as cropping the plot with specific range of axis values. Additionally, labels can be assigned for individual data points for higher verbosity. In this study, the graphs are analyzed as a whole, and due to the amount

of data points, no labels are used.

As mentioned in the previous section, each cluster will be colored differently on the graph. Additionally, in this case, the layout of the graph features legends for each cluster, which can be toggled to hide or show individual clusters on the chart. This allows for inspecting individual clusters as necessary.

Should a cluster be established as a user group, its legend can be given a distinct name for easier identification. Moreover, if new user groups are found, new visitors on the site can quickly be assigned into these groups by cross-referencing the attributes from the new user's session with the clustered results. As the implementation of the `K-means clustering` algorithm offers a method for achieving this in real time, new users can be assigned to a group and offered content targeted to their group as they enter the site.

5. RESULTS AND ANALYSIS

This chapter explores the results given both by the `K-means clustering` algorithm and the charts plotted by `plotly.js` both before and after the clustering of data. An analytic approach is taken towards the results to examine the validity of the method in terms of identifying existing groups of users as well as new users.

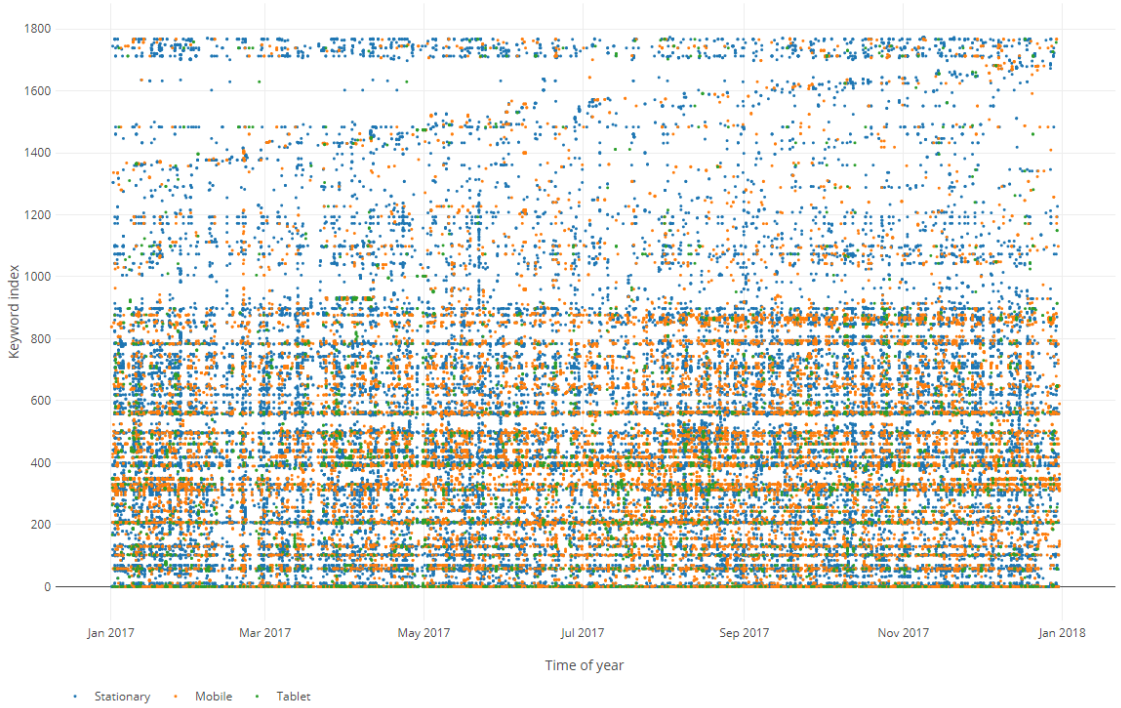
Additionally, some suggestions for further development of the study are given. The most relevant information in the source session data was chosen and its values cross-referenced in a bivariate scatterplot. Several samples were taken, including yearly traffic, snapshots of a single day as well as periods of one month layered onto a 24-hour timescale.

5.1 Validity of data for identification of user groups

Plotting the data without clustering it beforehand yields preliminary results, which can be observed and their validity estimated. In figures 5.1, 5.2, 5.3, 5.4 and 5.6 the keywords of the visited page have been plotted as a function of time. The order in which the keywords feature on the y axis is determined by their assigned relative website path, which is listed in descending alphabetical order. The charts also feature the type of device used to access the site as separate data sets, denoted with different colored traces. In this case, stationary computers have been colored blue, mobile devices orange and tablets green.

In figure 5.1, entries of the first thirty thousand (30000) results of the year 2017 are displayed. In reality, the website experiences millions of visits every year, but due to memory limitations of the browser, the amount was limited. The traces seem to be exhibiting areas of varying density along the y axis, signifying that certain keywords are more popular than others, and that the traffic is concentrated around certain pages on the website.

Figure 5.1 *Yearly traffic*

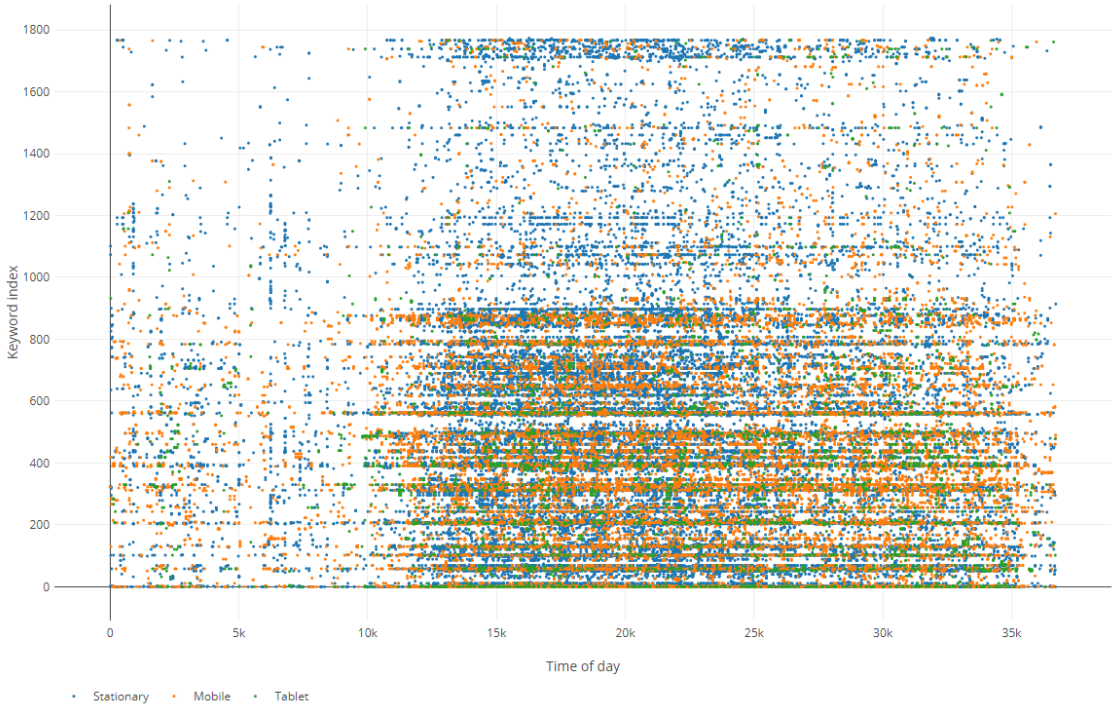


However, the data is spread uniformly on the x axis throughout the graph. Therefore, time of year does not seem to have any noticeable effect on the accessed content on the site. Some keywords seem to have no hits throughout a portion the year while exhibiting increased hits later, which could be explained by the amount of content varying throughout the year. Some websites, such as the one used in this study, may feature content management systems (CMS), which allows administrators and editors to edit the content of the site.

Figure 5.2 shows the same data set layered over a single day. Here the same vertical clustering around certain pages can be seen. In addition, a clear decrease in activity during the night can be seen, along with a more dense area during office hours. Due to the limitations of the `plotly.js` graphing library, the time of day could not be translated back to 24-hour format.

Beyond the decrease during the night and increase during office hours, website use does not seem to experience major fluctuation during the day. In addition to the heightened use during office hours, small drops in activity can be noticed during morning and afternoon commutes.

Figure 5.2 *Daily traffic*

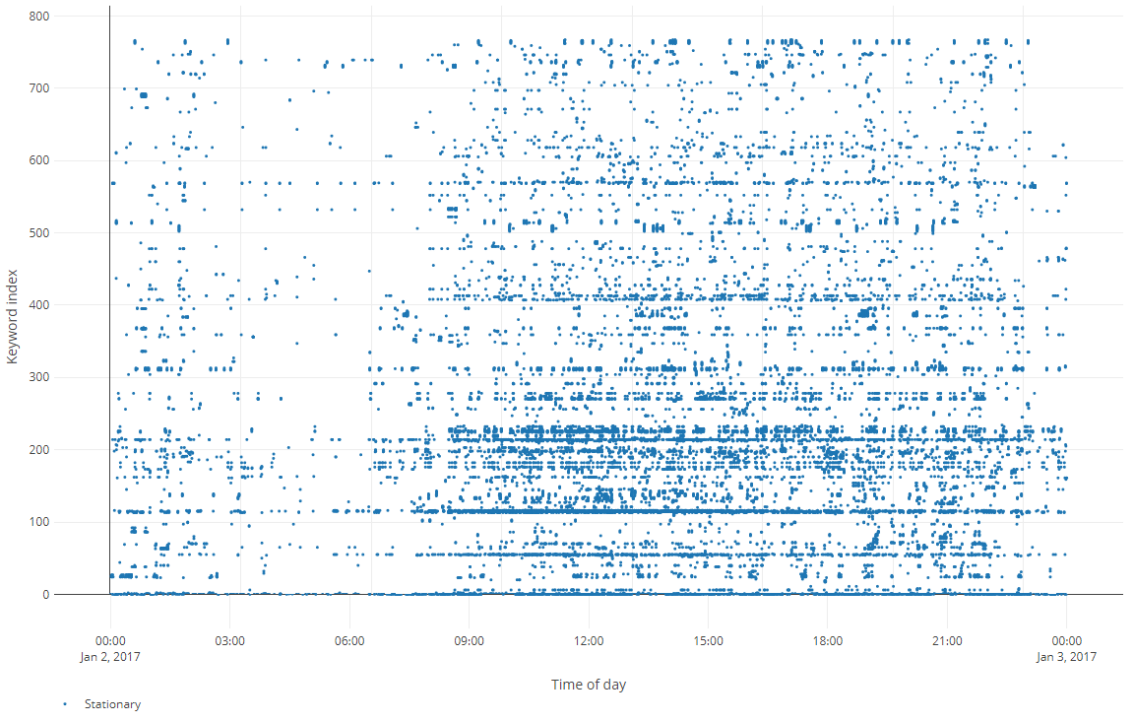


Different keywords seem to exhibit varying degrees of use for different devices. For example, some keywords exhibit steady mobile access throughout the afternoon and evening, despite stationary use being concentrated before and after noon. This could be caused by the ubiquity and ease of use of mobile devices compared to stationary computers.

Additionally, certain keywords have heightened use for stationary devices during the day, while seeing few mobile accesses. This could be attributed to the nature of the pages; for example, users may prefer to access certain content with stationary computers rather than mobile devices.

Figure 5.3 displays the keyword hits on the site from stationary devices on the first monday of January of year 2017. Here, the individual results are more discernable by the human eye, however there is still major overlapping around certain keywords during peak hours.

Figure 5.3 *Website accesses on stationary devices, 02.01.2017*

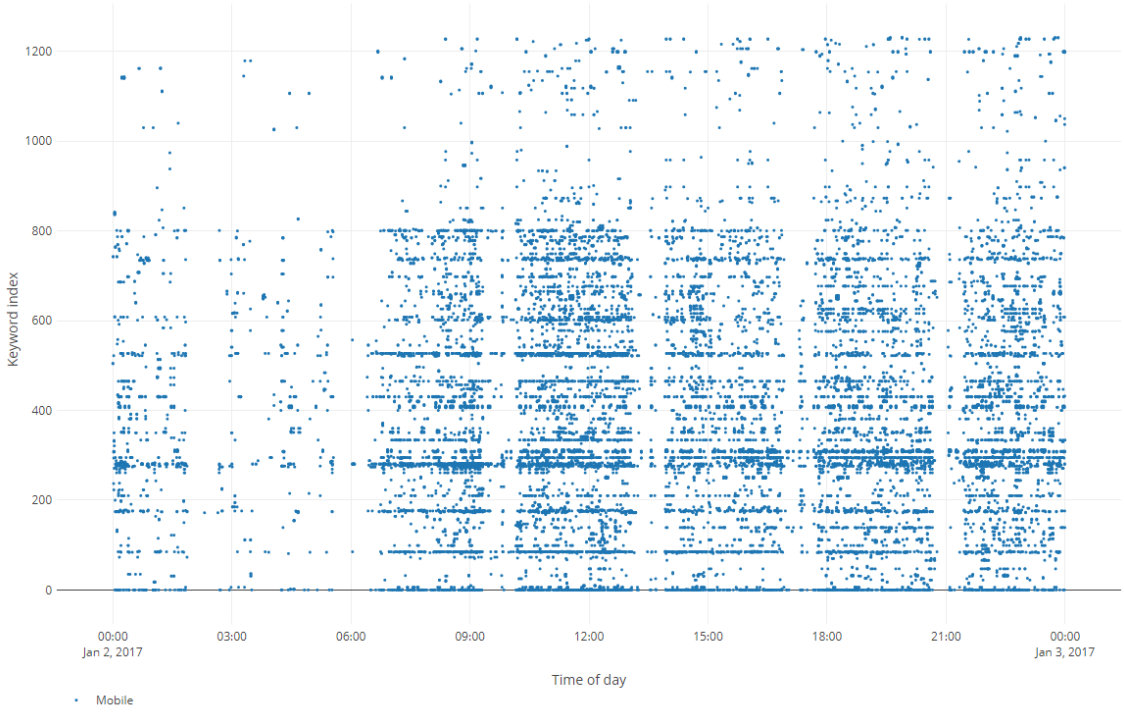


Here, the popularity of certain keywords can be seen more clearly. However, no major clusters can be found within the data, and the more sparse areas exhibit large amounts of noise.

Figure 5.3 also shows a decreased amount of keywords. As explained before, the amount of content on the website may vary throughout the year, resulting in an increased count in keywords in yearly traffic. Additionally, some content may not have been accessed on the chosen day.

Figure 5.4 displays the first 30000 keyword hits for the same day, but for mobile accesses. Attempting to raise the limit similarly caused the browser tab process to run out of memory. Additionally, the library seems to render the results in specific intervals on the x axis, resulting in sections with no data. Beyond the limited dataset, the access patterns for mobile do not seem to differ discernably from stationary users.

Figure 5.4 *Website accesses on mobile devices, 02.01.2017*



However, based on the number of keywords, mobile users seem to have access to a wider range of content. This could be caused by separate mobile versions of pages having different keywords. Additionally, mobile users seem to be more active than stationary users during the late hours.

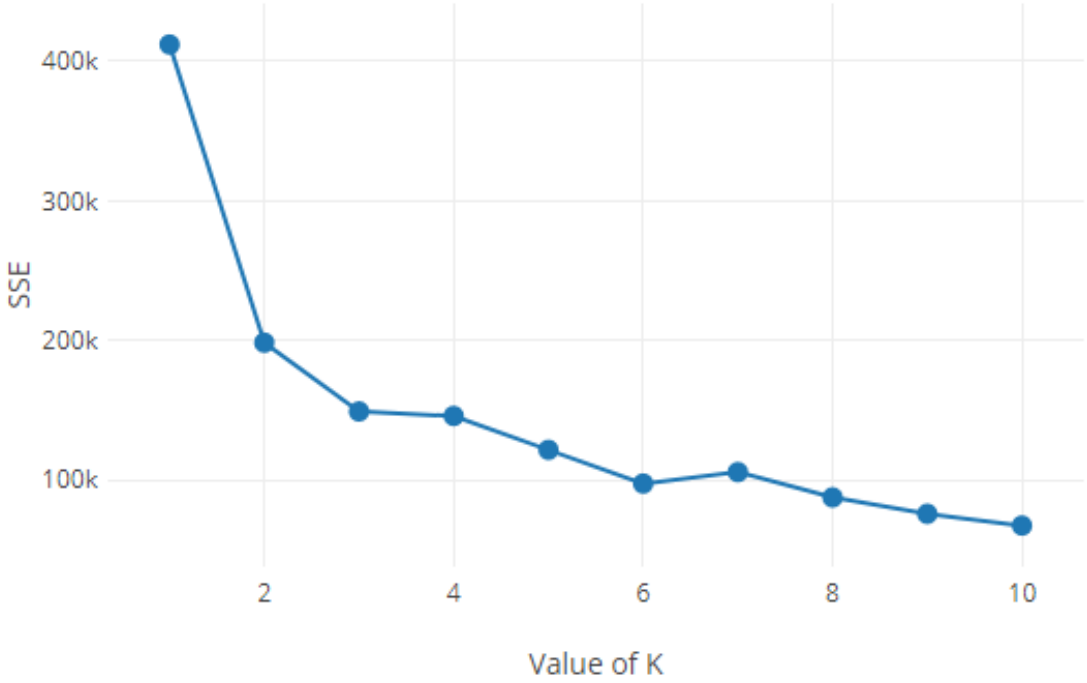
Due to the limited dataset, mobile access data is not valid to be clustered. Therefore, as the structure is similar, only stationary access is clustered and evaluated.

5.2 Effects of the clustering algorithm

As explained in section 4.2, the `K-means clustering` algorithm requires an initial value of K, which was chosen with the `elbow method` by examining the SSE. Additionally, the provided implementation of the algorithm on the *Apache Ignite* machine learning platform limits the values of K to a maximum of ten (10). Therefore, SSE was plotted for the first 10 values of K.

Due to different days exhibiting similar structures in their data sets, it is sufficient to examine the clustering of the visits of one day. In this case, the first monday of January 2017 was chosen. The `K-means clustering` algorithm was run once with this data set for every value of K from one (1) to ten (10), and the value of SSE was calculated for every value of K. The resulting values can be seen in figure 5.5.

Figure 5.5 *SSE with* $K_{max} = 10$

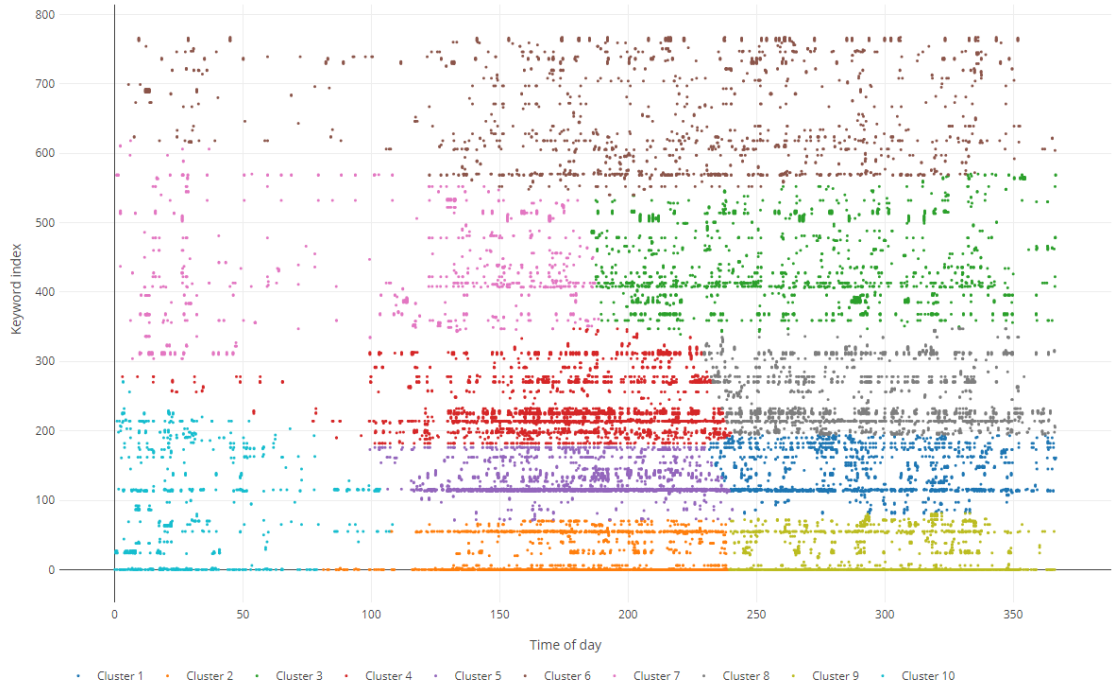


As seen in the graph, the sum of squares error does not form a clear elbow. The most significant difference in the margin is between $K = 1$ and $K = 2$, however as seen in the graphs in section 5.1, the value of K must be higher to provide meaningful clusters.

The next two candidates for the value of K are six (6) and ten (10). The margin between $K = 6$ and $K = 10$ is relatively small, however due to the high amount of different keywords and possible clusters, a high value of K is desired. Therefore, the value of K was determined to be ten (10).

Afterwards, the output of the algorithm with an initial value of $K = 10$ was plotted. Figure 5.6 displays the data set after the clustering, which resulted in several areas of data points. Each area is colored differently for easier recognition. Due to limitations in the graphing library with displaying custom timestamps, values on the x axis are denoted as seconds divided by one hundred (100).

Figure 5.6 *Clusterization of website accesses, 02.01.2017*



The resulting graph clearly displays smaller clusters where the heaviest traffic during office and evening hours is concentrated. Additionally, larger, more sparse clusters can be found around the less accessed keywords, however the borders of the clusters are unambiguous due to the horizontally uniform structure of the data. As seen from the chart, the `K-means clustering` algorithm performs poorly with the chosen data set, and therefore is suboptimal for producing meaningful results.

There are three fundamental issues with the results of the clusterization. First, the limitation of the value of K in the algorithm's implementation also limits the clustering of data with high amounts of data points spread across hundreds of variable values. Second, the data set exhibits a clustering structure only along one axis, and beyond the decrease of traffic during the night, the visits are spread uniformly over the time of day. Finally, there is too much intracluster noise between the denser areas for the clustering algorithm to perform well.

5.3 Further development

As shown by the studies by Li et al. [24] and especially Poornalatha et al. [31], anonymous session data can be used to produce meaningful results. Therefore, despite the issues described in section 5.2 and with sufficient research and development, the data used in this study could potentially provide insight to the browsing patterns of website visitors. However, this would require approaching the problem from

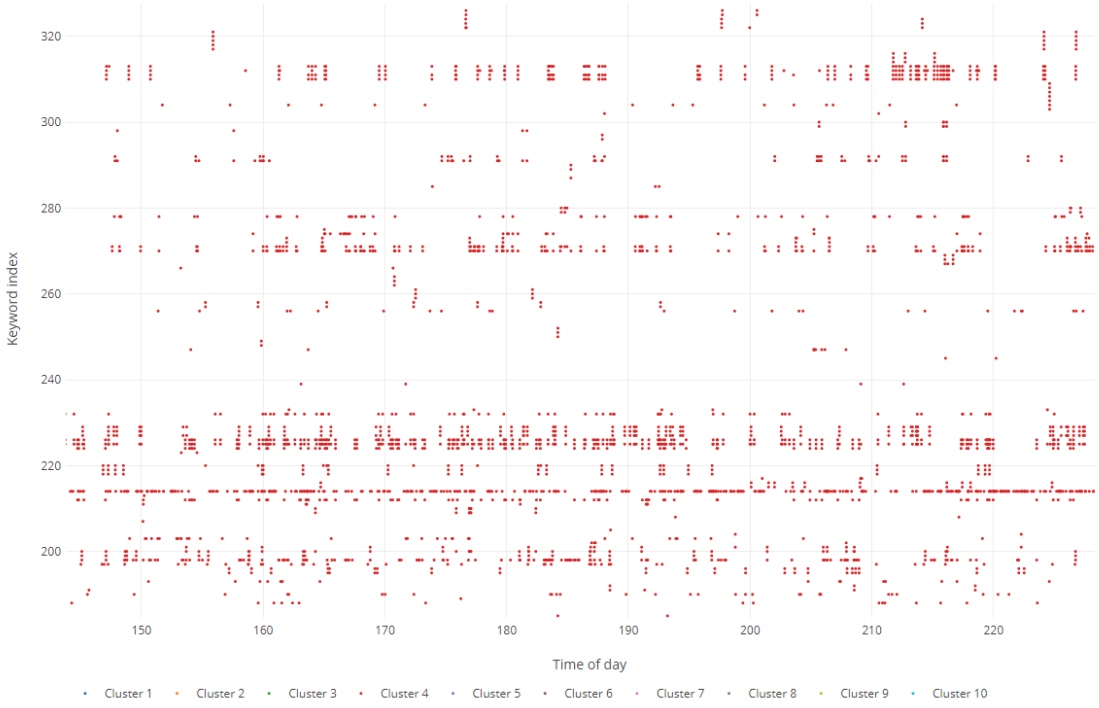
a different perspective and considering different methods of research.

The current method could be developed further by solving the aforementioned issues. This, however, would require the limitations of the value of K to be removed, so that the clustering could be done on a more detailed level. The maximum value of K is set in the Java class `KMeansTrainer`, which is imported and called upon in the example implementation used in this study, and changing it would require altering the source code of the class. Altering the source code, in turn, would require further development and testing, and therefore exceeds the scope of this study.

Alternatively, the clustering algorithm could be run with a smaller excerpt of the data. This would include either choosing a certain portion of the keywords while excluding the rest entirely, or limiting the examined hits to a specific time of day. For example, in this case, fifty (50) keywords with the most hits could be chosen. This would allow for examining whether the time of day has effect on specific keywords or keyword groups around certain relative paths.

Additionally, the time of day could be limited to certain intervals, providing a more detailed image specific times of day. For example, cropping most of cluster 4 in the results in figure 5.6, introduced in section 5.2, would result in a snapshot of some of the most popular keywords and their hits during peak office hours, as depicted in figure 5.7.

Figure 5.7 *Crop of cluster 4*



As seen from the resulting chart, some minor clustering can be seen, however the structure remains suboptimal for the `K-means clustering` algorithm. Additionally, clustering the data set in cropped sections would require further development and could result in complications with the implementation as well as finding a suitable method for dividing the data set into meaningful sections.

Finally, a different algorithm could be used. With a data set with a lot of noise and no clear clusters, such as that used in this study, a different algorithm could provide more meaningful results. Alternatives to the `K-means clustering` algorithm include density-based algorithms such as density-based spatial clustering applications with noise (DBSCAN) [40] as well as Gaussian mixture model (GMM) -based methods [38].

The DBSCAN algorithm, as its name implies, is suitable for data sets that exhibit variations in data density and a degree of intercluster noise. As explained by Wang et al. [40], the algorithm defines clusters based on matching a minimum number of points within a given radius of any point in the data set. Gaussian mixture models are based on probability functions of random variables that follow the Gaussian distribution [14]. According to Garcia et al., GMMs are a crucial tool in statistical modeling for more challenging applications in computer vision and machine learning.

The Apache Ignite platform does not offer algorithm implementations of DBSCAN or GMM, however, as it is implemented with Java, it can employ the Scala API of the Apache Spark platform [11]. Apache Spark is an analytics engine for large-scale data processing by the Apache Foundation [13] that has libraries for machine learning and clustering, including Gaussian mixture models.

6. CONCLUSIONS

With the ever increasing amount of information, data and services on the Internet, the importance of delivering service users relevant content on websites is emphasized from both a user experience as well as a business perspective. Due to this, recommender systems have been employed by content-heavy websites to automate the process of identifying, profiling and providing suggestions relevant to users' interests.

However, these systems require statistical data to function. This study explored the option of utilizing machine learning and cluster analysis to examine ubiquitous and anonymous session data as a basis for providing targeted content and a more lightweight alternative to recommender systems. The session data used in the study was visualized before and after clustering to examine the presence of clusters meaningful enough to be considered a group of users.

The study was conducted by employing the open source platforms of Apache Solr and Apache Ignite. The website visitor tracking system from which the session data used in the study was retrieved is implemented on the Apache Solr platform. Apache Ignite was chosen due to its distributed computing capabilities and ready access to clustering algorithm implementations.

The K-means clustering algorithm was chosen due to its popularity and lightweight implementation, however the results show it to perform suboptimally with the data used. The algorithm requires a clear clustering structure to be present in the data, however due to the high amount of intracluster noise, no distinct clusters could be found. In addition, the implementation of the algorithm used in the study used a limited initial value of K. This caused further issues with the accuracy of the clustering in a large-scale data set. Furthermore, the graphing library chosen for the study was shown to perform poorly with large-scale data sets, limiting the visual analysis.

Despite these issues, the visualization shows areas of density and sparsity in the data. This means the subject could be studied further with a different method. Further

studies could be made utilizing a algorithm, such as the density-based DBSCAN algorithm. Another approach could be to further develop the implementation of the K-means clustering algorithm to remove the current limitations for clustering the data on a more detailed level.

While this method does not allow for identifying user groups for targeted content, it shows potential for a lightweight solution for quickly identifying new users on a website. Session and website access log data is pervasive and universal and holds information that, if parsed and studied correctly, can be used to identify and classify website users' interests and browsing patterns.

BIBLIOGRAPHY

[1] *Website Hosting and Website Management*. London: Springer London, 2007, pp. 221–231. [Online]. Available: https://doi.org/10.1007/978-1-84628-795-4_10

[2] A. Barth, "HTTP State Management Mechanism," Internet Requests for Comments, Internet Engineering Task Force (IETF), RFC 6265, April 2011. [Online]. Available: https://tools.ietf.org/html/rfc6265

[3] T. Berners-Lee, "Hypertext Markup Language - 2.0," Internet Requests for Comments, Internet Engineering Task Force (IETF), RFC 6265, November 1995. [Online]. Available: https://tools.ietf.org/html/rfc1866

[4] M. Bostock, "Data-driven documents," https://d3js.org, 2017, Accessed: 02.10.2018.

[5] S. P. Chatzis, V. Siakoulis, A. Petropoulos, E. Stavroulakis, and N. Vlachogiannakis, "Forecasting stock market crisis events using deep and statistical machine learning techniques," *Expert Systems with Applications*, vol. 112, pp. 353 – 371, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417418303798

[6] R. Colombo, "Visitor patterns of a profitable website," *Interactive Marketing*, vol. 3, no. 1, pp. 30–41, Jul 2001. [Online]. Available: https://doi.org/10.1057/palgrave.im.4340110

[7] I. D. Dinov, *k-Means Clustering*. Cham: Springer International Publishing, 2018, pp. 443–473. [Online]. Available: https://doi.org/10.1007/978-3-319-72347-1_13

[8] R. Fernandes de Mello and M. Antonelli Ponti, *A Brief Review on Machine Learning*. Cham: Springer International Publishing, 2018, pp. 1–74. [Online]. Available: https://doi.org/10.1007/978-3-319-94989-5_1

[9] S. Flesca, S. Greco, A. Tagarelli, and E. Zumpano, "Mining user preferences, page content and usage to personalize website navigation," *World Wide Web*, vol. 8, no. 3, pp. 317–345, Sep 2005. [Online]. Available: https://doi.org/10.1007/s11280-005-1315-9

[10] C. for Machine Learning and I. Intelligent Systems, University of California, "Iris Data Set," https://archive.ics.uci.edu/ml/datasets/iris, Accessed: 17.09.2018.

[11] A. S. Foundation, "Apache Ignite," https://ignite.apache.org/whatisignite.html, Accessed: 07.09.2018.

[12] ——, "Apache Solr," http://lucene.apache.org/solr/features.html, Accessed: 01.09.2018.

[13] ——, "Apache Spark," https://spark.apache.org, Accessed: 11.10.2018.

[14] V. Garcia, F. Nielsen, and R. Nock, "Levels of details for gaussian mixture models," in *Computer Vision – ACCV 2009*, H. Zha, R.-i. Taniguchi, and S. Maybank, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 514–525.

[15] M. L. Giger, "Machine learning in medical imaging," *Journal of the American College of Radiology*, vol. 15, no. 3, Part B, pp. 512 – 520, 2018, data Science: Big Data Machine Learning and Artificial Intelligence. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1546144017316733

[16] C. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Trans. Manage. Inf. Syst.*, vol. 6, no. 4, pp. 13:1–13:19, Dec. 2015. [Online]. Available: http://doi.acm.org/10.1145/2843948

[17] P. Hansen, B. Jaumard, and N. Mladenovic, "Minimum sum of squares clustering in a low dimensional space," *Journal of Classification*, vol. 15, no. 1, pp. 37–55, Jan 1998. [Online]. Available: https://doi.org/10.1007/s003579900019

[18] W. K. Härdle and L. Simar, *Cluster Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 331–349. [Online]. Available: https://doi.org/10.1007/978-3-642-17229-8_12

[19] R. H. Ip, L.-M. Ang, K. P. Seng, J. Broster, and J. Pratley, "Big data and machine learning for crop protection," *Computers and Electronics in Agriculture*, vol. 151, pp. 376 – 383, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169917314588

[20] T. W. Jackson, "Personalisation and crm," *Journal of Database Marketing & Customer Strategy Management*, vol. 15, no. 1, pp. 24–36, Oct 2007. [Online]. Available: https://doi.org/10.1057/palgrave.dbm.3250065

[21] S. C. James, Y. Zhang, and F. O'Donncha, "A machine learning framework to forecast wave conditions," *Coastal Engineering*, vol. 137, pp. 1 – 10, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378383917304969

[22] X. Jin and J. Han, *K-Means Clustering*. Boston, MA: Springer US, 2010, pp. 563–564. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_425

[23] B.-R. Lea, W.-B. Yu, and H. Min, "Data visualization for assessing the biofuel commercialization potential within the business intelligence framework," *Journal of Cleaner Production*, vol. 188, pp. 921 – 941, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0959652618306267

[24] X. Li and Y. Cheng, "The improved clustering algorithm for mining user's preferred browsing paths," in *Intelligent Computing Methodologies*, D.-S. Huang, K. Han, and A. Hussain, Eds. Cham: Springer International Publishing, 2016, pp. 329–337.

[25] F. Masulli and S. Rovetta, "Clustering high-dimensional data," in *Clustering High–Dimensional Data*, F. Masulli, A. Petrosino, and S. Rovetta, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 1–13.

[26] Mozilla, "Blob," https://developer.mozilla.org/en-US/docs/Web/API/Blob, 2018, Accessed: 15.10.2018.

[27] A. Neumann, *Scalable Vector Graphics (SVG)*. Cham: Springer International Publishing, 2017, pp. 1831–1840. [Online]. Available: https://doi.org/10.1007/978-3-319-17885-1_1159

[28] I. O. Pappas, P. E. Kourouthanassis, M. N. Giannakos, and V. Chrissikopoulos, "Shiny happy people buying: the role of emotions on personalized e-shopping," *Electronic Markets*, vol. 24, no. 3, pp. 193–206, Sep 2014. [Online]. Available: https://doi.org/10.1007/s12525-014-0153-y

[29] Plotly, "plotly.js," https://plot.ly/javascript/reference/, 2018, Accessed: 01.10.2018.

[30] ——, "plotly.js," https://github.com/plotly/plotly.js, 2018.

[31] G. Poornalatha and P. S. Raghavendra, "Web user session clustering using modified k-means algorithm," in *Advances in Computing and Communications*, A. Abraham, J. Lloret Mauri, J. F. Buford, J. Suzuki, and S. M. Thampi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 243–252.

[32] I. Portugal, P. Alencar, and D. Cowan, "The use of machine learning algorithms in recommender systems: A systematic review," *Expert Systems with Applications*, vol. 97, pp. 205 – 227, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417417308333

[33] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, p. 131, Dec 2008. [Online]. Available: https://doi.org/10.1007/s10664-008-9102-8

[34] R. Sullivan, *Machine-Learning Techniques*. Totowa, NJ: Humana Press, 2012, pp. 363–454. [Online]. Available: https://doi.org/10.1007/978-1-59745-290-8_8

[35] M. Syakur, B. Khotimah, E. Rochman, and B. Satoto, "Integration k-means clustering method and elbow method for identification of the best customer profile cluster," vol. 336, no. 1, 2018. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85046287552&doi=10.1088%2f1757-899X%2f336%2f1%2f012017&partnerID=40&md5=ef773be5853e1ccfa021375894c727e0

[36] P. Symeonidis, D. Ntempos, and Y. Manolopoulos, *Recommender Systems*. New York, NY: Springer New York, 2014, pp. 7–20. [Online]. Available: https://doi.org/10.1007/978-1-4939-0286-6_2

[37] M. Tanaka, "c3.js reference," https://c3js.org/reference.html, 2018, Accessed: 12.10.2018.

[38] C.-Y. Tsai and C.-C. Chiu, "An efficient feature selection approach for clustering: Using a gaussian mixture model of data dissimilarity," in *Computational Science and Its Applications – ICCSA 2007*, O. Gervasi and M. L. Gavrilova, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1107–1118.

[39] J. Turnbull, *Linux Basics*. Berkeley, CA: Apress, 2009, pp. 63–108. [Online]. Available: https://doi.org/10.1007/978-1-4302-1913-2_3

[40] X. Wang and H. J. Hamilton, "A comparative study of two density-based spatial clustering algorithms for very large datasets," in *Advances in Artificial Intelligence*, B. Kégl and G. Lapalme, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 120–132.

[41] Z. Wang, Q. Qi, and L. Xu, "Cluster analysis based on spatial feature selecting in spatial data mining," in *2008 International Conference on Computer Science and Software Engineering*, vol. 4, Dec 2008, pp. 386–389.

[42] J. Wu, *Cluster Analysis and K-means Clustering: An Introduction*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–16. [Online]. Available: https://doi.org/10.1007/978-3-642-29807-3_1

[43] S. Yong, S. Zhen-Chao, Z. Ran, Z. Geng, and L. Shi-Dong, "An improved cluster analysis algorithm using for network traffic flow," in *2015 10th International Conference on Computer Science Education (ICCSE)*, July 2015, pp. 111–115.

APPENDIX A. APACHE SOLR CORE EXTRACT


```
{
    "UA.Browser.Name": "Firefox",
    "UserModified": 1516793180762,
    "HitDayOfMonth": 22,
    "HitHour": 15,
    "HitId": "a3NfqEroS",
    "UserCreated": 1453277678425,
    "UA.OS.Name": "Android",
    "UA.Engine.Name": "Gecko",
    "HitSecond": 28,
    "UA.Device.Model": "",
    "HitTime": 1461328588275,
    "UserAgent": "Mozilla/5.0 (Android 5.0; Mobile; rv:45.0) Gecko/45.0
        ↪ Firefox/45.0",
    "STATO_PROTECTION": [
        "NotProtected"
    ],
    "STATO_FULLPATH": "/channels/public/www/vesi/fi/index",
    "UA.CPU.Architecture": "",
    "SessionId": "a3NeAPgSC",
    "SessionLastHit": 1461328588275,
    "HostName": "85-76-134-9-nat.elisa-mobile.fi",
    "Path": "/Tampere_5/channels/public/www/vesi/fi/index",
    "HostIp": "85.76.134.9",
    "UA.OS.Version": "5.0",
    "UA.Browser.Major": "45",
    "UA.Device.Vendor": "",
    "UA.Browser.Version": "45.0",
    "UA.Engine.Version": "45.0",
    "DomainName": "elisa-mobile.fi",
    "Referer": "http://tampere58www-test.abako.fi/vesi/index.html",
    "HitMinute": 36,
    "HitYear": 113,
    "HitMonth": 4,
    "Date": "2016-04-22T12:36:28.275Z",
    "HitWeekday": 5,
    "HitWeekOfYear": 16,
    "HitDate": "2016-04-22 15:36:28",
    "SessionFirstHit": 1461328588275,
    "UserId": "anonymous",
    "UA.Device.Type": "mobile",
    "_version_": 1591550325307736000
}
```

APPENDIX B. APACHE IGNITE K-MEANS CLUSTERING EXAMPLE

```
/*
 * Licensed to the Apache Software Foundation (ASF) under one or more
 * contributor license agreements. See the NOTICE file distributed with
 * this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You under the Apache License, Version 2.0
 * (the "License"); you may not use this file except in compliance with
 * the License. You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package org.apache.ignite.examples.ml.clustering;

import java.util.Arrays;
import java.util.UUID;
import javax.cache.Cache;
import org.apache.ignite.Ignite;
import org.apache.ignite.IgniteCache;
import org.apache.ignite.Ignition;
import org.apache.ignite.cache.affinity.rendezvous.RendezvousAffinityFunction;
import org.apache.ignite.cache.query.QueryCursor;
import org.apache.ignite.cache.query.ScanQuery;
import org.apache.ignite.configuration.CacheConfiguration;
import org.apache.ignite.ml.dataset.impl.cache.CacheBasedDatasetBuilder;
import org.apache.ignite.ml.knn.classification.KNNClassificationTrainer;
import org.apache.ignite.ml.math.Tracer;
import org.apache.ignite.ml.math.distances.EuclideanDistance;
import org.apache.ignite.ml.math.distances.DistanceMeasure;
import org.apache.ignite.ml.math.impls.vector.DenseLocalOnHeapVector;
import org.apache.ignite.ml.clustering.kmeans.KMeansModel;
import org.apache.ignite.ml.clustering.kmeans.KMeansTrainer;
import org.apache.ignite.thread.IgniteThread;
import java.util.ArrayList;
import java.util.List;
import java.lang.Double;
```

```
import java.nio.file.Paths;
import java.nio.file.Files;
import java.io.IOException;
import java.util.regex.Pattern;
import java.io.FileWriter;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

/**
 * Run kNN multi-class classification trainer over distributed dataset.
 *
 * @see KNNClassificationTrainer
 */
public class KMeansClusterizationExample {
    /** Run example. */
    public static void main(String[] args) throws InterruptedException {
        System.out.println();
        System.out.println(">>> KMeans clustering algorithm over cached
            ↪ dataset usage example started.");

        int K_value = Integer.parseInt(args[0]);
        System.out.println(">>> " + K_value);

        // Start ignite grid.
        try (Ignite ignite = Ignition.start("examples/config/example-ignite.
            ↪ xml")) {
            System.out.println(">>> Ignite grid started.");

            // Start ignite thread
            IgniteThread igniteThread = new IgniteThread(ignite.configuration()
                ↪ .getIgniteInstanceName(),
                KMeansClusterizationExample.class.getSimpleName(), () -> {
                IgniteCache<Integer, double[]> dataCache = getTestCache(ignite)
                    ↪ ;

                // KMeans trainer, input seed and number of cluster centroids
                KMeansTrainer trainer = new KMeansTrainer()
                    .withSeed(7867L)
                    .withK(K_value);

                KMeansModel mdl = trainer.fit(
                    new CacheBasedDatasetBuilder<>(ignite, dataCache),
                    (k, v) -> Arrays.copyOfRange(v, 1, v.length),
                    (k, v) -> v[0]
                );

                // Print out cluster centroids
                System.out.println(">>> KMeans centroids");
                for (int i = 0; i < K_value; i++) {
                    Tracer.showAscii(mdl.centers()[i]);
                }
```

```
System.out.println(">>>");

System.out.println(">>> ---------------------------------");
System.out.println(">>> | Cluster #\t| Coordinates\t\t\t|");
System.out.println(">>> ---------------------------------");

int amountOfErrors = 0;
int totalAmount = 0;

 // Initialize distance variable for calculating SSE
EuclideanDistance distance = new EuclideanDistance();

// Map of cluster centroids and the distances to its datapoints
List<List<Double>> clusterMap = new ArrayList<List<Double>>();
for (int i = 0; i < K_value; i++) {
    clusterMap.add(new ArrayList<Double>());
}

// Arraylists for output
List<List<Double>> xAxis = new ArrayList<List<Double>>();
for (int i = 0; i < K_value; i++) {
    xAxis.add(new ArrayList<Double>());
}
List<List<Double>> yAxis = new ArrayList<List<Double>>();
for (int i = 0; i < K_value; i++) {
    yAxis.add(new ArrayList<Double>());
}

// Calculate predictions and actual labels of dataset entries
try (QueryCursor<Cache.Entry<Integer, double[]>> observations =
    ↪  dataCache.query(new ScanQuery<>())) {
    for (Cache.Entry<Integer, double[]> observation :
        ↪ observations) {
        double[] val = observation.getValue();
        double[] inputs = Arrays.copyOfRange(val, 1, val.length)
            ↪ ;
        double groundTruth = val[0];

        // Assigning data points to clusters
        double prediction = mdl.apply(new DenseLocalOnHeapVector
            ↪ (inputs));

        // Calculate distance between datapoint and its assigned
            ↪  cluster centroid
        int clusterIndex = new Double(prediction).intValue();
        double ret = distance.compute(mdl.centers()[clusterIndex
            ↪ ],inputs);

        // Assign distance to the clustermap
        clusterMap.get(clusterIndex).add(new Double(ret));
```

```
            totalAmount++;
            if (groundTruth != prediction)
                amountOfErrors++;

            // Output arrays
            xAxis.get(clusterIndex).add(new Double(inputs[0]));
            yAxis.get(clusterIndex).add(new Double(inputs[1]));

        }

        try (FileWriter xAxisFile = new FileWriter("C:/Users/ahti/
            ↪ git/kmeansclustering/elements/graph/js/xAxis.js")) {
            for (int i = 0; i < xAxis.size(); i++) {
                xAxisFile.write("var xAxis"+i+" = "+xAxis.get(i)+";\
                    ↪ n");
            }
        } catch(IOException e) {
            e.printStackTrace();
        }
        try (FileWriter yAxisFile = new FileWriter("C:/Users/ahti/
            ↪ git/kmeansclustering/elements/graph/js/yAxis.js")) {
            for (int i = 0; i < yAxis.size(); i++) {
                yAxisFile.write("var yAxis"+i+" = "+yAxis.get(i)+";\
                    ↪ n");
            }
        } catch(IOException e) {
            e.printStackTrace();
        }

        // Calculating SSE
        double sum = 0;
        double average;
        double[] sse = new double[K_value];

        for (int i = 0; i < clusterMap.size(); i++) {
            sum = 0;
            sse[i] = 0;

            for (double d : clusterMap.get(i)) sum += d;
            average = sum / clusterMap.get(i).size();

            for (int j = 0; j < clusterMap.get(i).size(); j++) {
                sse[i] += Math.pow(clusterMap.get(i).get(j) -
                    ↪ average, 2);
            }
            System.out.println("Cluster "+i+" SSE: "+sse[i]);
        }
```

```
                    System.out.println(">>> ---------------------------------");

                    System.out.println("\n>>> Absolute amount of errors " +
                        ↪ amountOfErrors);
                    System.out.println("\n>>> Accuracy " + (1 - amountOfErrors /
                        ↪  (double)totalAmount));
                }
            });

            igniteThread.start();
            igniteThread.join();
        }
    }
    /**
     * Fills cache with data and returns it.
     *
     * @param ignite Ignite instance.
     * @return Filled Ignite Cache.
     */
    private static IgniteCache<Integer, double[]> getTestCache(Ignite ignite)
        ↪ {

        double[][] data = new double[0][0];
        Pattern spacePattern = Pattern.compile(" ");
        try {
            data = Files.lines(Paths.get("/* Path to file */data.txt"))
                .map(item -> spacePattern.splitAsStream(item).mapToDouble(
                    ↪ Double::parseDouble).toArray())
                .toArray(double[][]::new);
        }
            catch(IOException e) {
            e.printStackTrace();
        }

        CacheConfiguration<Integer, double[]> cacheConfiguration = new
            ↪ CacheConfiguration<>();
        cacheConfiguration.setName("TEST_" + UUID.randomUUID());
        cacheConfiguration.setAffinity(new RendezvousAffinityFunction(false,
            ↪ 10));

        IgniteCache<Integer, double[]> cache = ignite.createCache(
            ↪ cacheConfiguration);
        for (int i = 0; i < data.length; i++)
            cache.put(i, data[i]);

        return cache;
    }
}
```